A Low-Cost Burn-In Tester Architecture to supply Effective Electrical Stress

A Low-Cost Burn-In Tester Architecture to Supply Effective Electrical Stress

Francesco Angione, *Student Member, IEEE*, Davide Appello, Paolo Bernardi, *Senior Member, IEEE*, Claudia Bertani, Giovambattista Gallo, Stefano Littardi, Giorgio Pollaccia, Walter Ruggeri, Matteo Sonza Reorda, *Fellow, IEEE*, Vincenzo Tancorre, and Roberto Ugioli

Abstract—Burn-In test equipment usually owns extensive memory capabilities to store pre-computed patterns to be applied to the circuit inputs as well as ad-hoc circuitries to drive and read the DUT pins during the BI phase. The solution proposed in this paper dramatically reduces the memory size requirement and just demands a generic microcontroller unit (MCU) equipped with a couple of embedded processors, some standard common peripheral units, and a few KB memories. Moreover, the proposed Burn-In tester could be integrated into a System Level Test equipment which is typically based on MCUs to communicate functionally with the DUT. This paper provides full details about the architecture of such a low-cost innovative tester, which can supply the DUT with unlimited pseudo-random patterns created autonomously by the MCU firmware from any selected seed. The tester prototype developed to collect experimental results includes a low-cost System-on-Chip based on a multi-core MCU and a set of peripheral cores, encompassing timers and Direct Memory Access modules. The tester prototype is used to stress an automotive chip accounting for about 20 million gates, 700 thousand scan flip-flops, and several scan modes. The combination of pseudo-random pattern generation with the ability to control different scan and Design for Testability (DfT) modes, including LBIST, permits to reach a higher coverage of stress metrics than by the application of a limited set of pre-computed ATPG patterns. The toggle coverage level reached is up to 95.89%. The application speed achieved by the tester with non-optimized connections is up to about 10MHz.

Index Terms—Reliability, testing, test generation, firmware engineering

1 Introduction

New trends are currently transforming high-volume manufacturing test in the semiconductor industry targeting safety-critical [1] products:

- The growing complexity of the manufactured Integrated Circuits (ICs), which often correspond to Systems-on-Chip (SoCs) composed of numerous Intellectual Property cores (IPs).
- The explosion in the number of advanced electronic devices and systems in safety-critical applications (e.g., ADAS in automotive applications).
- The growing awareness of failure modes detectable only through a functional activation in many cases.

These factors reflect into a growing complex test equipment scenario, demanding memory capabilities and frequency requirements as well as application conditions, such as temperature and voltage margins. Traditionally, the Final Test (FT) is based on structural tests generated by an Automatic Test Pattern Generator (ATPG) targeting high coverage for the considered fault models. Today, especially for automotive devices, the Final Test is often considered insufficient to achieve the market's expected quality levels. Additional steps are necessary, including Burn-In (BI) and System Level Test (SLT). The purpose of the Burn-In process [1] is to activate infant mortality (early life latent defects) that naturally affects populations of electronic devices. Burn-In is a manufacturing test phase used for many mission-critical modules, such as automotive [2] microcontrollers and SoCs, which are the objective of the present research. In this field, BI is crucial for meeting the constraints coming from safety standards such as IEC 61805 [3] and ISO 26262 [4].

A BI tester applies two types of stress. The former is the *External stress*, which provides a high temperature and a different voltage margin than user-mode [1]. This kind of stress is supplied through a climatic chamber, which warms the chips up to their specification limits, and by using tunable voltage regulators mounted on the driver part of the test equipment, thus bringing the device around its voltage margins. This type of stress is directly related to Arrhenius's law about material aging. The latter is called *Internal stress*, which is produced by activating the functionalities of the devices during the BI phase. The main idea of BI is to combine External and Internal stress to accelerate the activation of extrinsic defects under the bathtub curve hypothesis [5].

The BI step has a high cost due to its long duration (up to 12 hours [6]). The cost of BI is managed through high parallelism, obtained by inserting the Devices Under Test

- *Francesco Angione, Paolo Bernardi, Giovambattista Gallo, Stefano Littardi, Walter Ruggeri, and Matteo Sonza Reorda are with Politecnico di Torino, Torino 10129, Italy. E-mail: {francesco.angione, paolo.bernardi, giovambattista.gallo, stefano.littardi, walter.ruggeri, matteo.sonzareorda}@polito.it.*
- *Davide Appello, Claudia Bertani, Giorgio Pollaccia, Vincenzo Tancorre, and Roberto Ugioli are with STMicroelectronics, Agrate Brianza 20864, Italy. E-mail: {davide.appello, claudia.bertani, giorgio.pollaccia, vincenzo.tancorre, roberto.ugioli}@st.com.*

Manuscript received 18 August 2021; revised 10 May 2022; accepted 6 June 2022. Date of publication 18 August 2022; date of current version 7 April 2023.
(Corresponding author: Paolo Bernardi.)
Recommended for acceptance by M. Kaaniche.
Digital Object Identifier no. 10.1109/TC.2022.3199994

(DUTs) on special boards, each hosting up to about one hundred DUTs. Nevertheless, parallelism maximization to contain BI costs also limits the BI equipment capabilities in terms of resource availability per DUT and power control [7].

System-Level Test (SLT) [8] is a functional test step that complements the common structural and parametric test steps. In the last years, several semiconductor companies included SLT as a final step in the test flow for GPUs [1] and mobile processor devices [9]. SLT is typically a test step where the target IC is placed in an environment as close as possible to the operational mode of the final application. Stimuli also mimic the functional ones, while the DUT behavior is compared with the expected one. SLT testers usually correspond to low-cost equipment where test stimuli generation and response management are jointly performed by DUT and some companion chips, e..g., a microcontroller unit controlling functional communication ports of the DUT. A major reason for including SLT in the manufacturing test flow lies in the growing gap between the behavior of devices in test conditions and the one in operational conditions. For example, the BIST procedures executed during the boot sequence at power-on might not correlate well with the activation of the same BIST procedures in test mode. The surrounding conditions (e.g., transients) and the influence of other IPs are usually the root causes for the non-complete correlation. Unfortunately, a logic simulator cannot simulate the power-on logic, and an Automated Test Equipment (ATE) is unlike to handle the power-on self-test logic. The increasing usage of serial and parallel high-speed interfaces is another critical point when resorting to a traditional ATE. The external loopback strategy is often adopted to reach full coverage, but HW and environment limitations expose the ATE step to the risk of high overkill rates.

This paper proposes a flexible tester architecture based on a low-cost multicore microcontroller unit (MCU) SoC able to provide effective internal electrical stress to Automotive DUTs. The firmware run by the MCU can generate on-the-fly pseudo-random patterns, use off-line created patterns, or make a combination of them. These patterns are stored in the SoC's embedded memories, and then sent to the scan chains of the DUT under the control of the MCU peripherals, such as Direct Memory Access (DMA), timers, and SPI units.

The online generation of stress patterns allows a massive reduction of memory footprint in the tester. Furthermore, it allows for storing multiple stress patterns, ranging from selective full scan ATPG patterns to LBIST-oriented patterns. This allows providing complete stress able to exacerbate potential latent defects before the final tests, e.g., System Level Test [10].

This approach looks in the imperative direction of test equipment optimization. The proposed methodology allows performing the Burn-In oriented internal stress procedure in other test steps, such as applying it along with the System Level Test phase [10]. The MCU-based tester stimulates scan ports as BI testers do with specialized architectures [11][12],. Thus, it enables the integration of these test phases and provides a significant cost-saving in test equipment and a decrease in the number of chip insertions during manufacturing flow. Nevertheless, it raises new issues, such as the availability of large channel memories typical of pure Burn-In equipment. As described in [10], the sustainability of the approach is directly related to the MCU cost and their re-usability for different DUTs. Cheap architectures are preferable, but they do not provide enough storage resources; therefore, a major challenge stands in reducing memory demands.

The paper describes how to overcome such strong memory limitations by the innovative use of a multicore MCU to generate by software a pseudo-random bitstream of unlimited length. This bitstream is then shifted in the scan chain(s) of a DUT at a reasonable speed (comparable with the usual BI communication frequency, up to about 10MHz). In the proposed tester organization, the pseudo-random stream produced by the firmware is temporarily stored into embedded RAMs before being sent to the input pins of the DUT. In order to effectively stimulate the DUT, the tester must be able to quickly generate the pseudo-random patterns and send them to the scan chain. In such a way, the RAM never overflows.

The ability to generate an endless stream of bits by software permits to apply many more patterns than a traditional BI tester could do. Based on the proposed tester's abilities, we compare the metric stress coverage reached by our tester with the one provided by the limited set of ATPG patterns typically used during regular BI. The pseudo-random stream provides a quite high but still incomplete stress coverage; for this reason, we have also evaluated how to obtain a further increase of coverage by going in the following directions:

- Resorting to the pseudo-random activation of internal resources like test data compression [13] engines, addressing those domains that the usual Burn-In scan chain-based stress cannot reach.
- Applying a limited number of selected ATPG-generated patterns for addressing the missing coverage and fitting the microcontroller's available memory resources.

The proposed technique enhances the traditional ATPG-based generation methodology. In particular, it enriches the purely ATPG-generated pattern in two ways:

- It greatly decreases the memory requirements (from many MBs to few KBs).
- It provides an internal stress ability, measured via several metrics, comparable to the current electrical stress achieved with a BI tester.

In the experimental part of the paper, we show the results obtained on a device for the automotive market, including around 20 million gates, implementing a full scan Design for Testability strategy involving around 700,000 Flip-Flops. The proposed approach reaches 92.18% of toggle activity coverage and 81.0% for the neighborhood-oriented static stress metric [14], described in Section 2.2, by applying 1,024 pseudo-random patterns. This value can be compared with the 90.73% and 76.97% figures, respectively, reached by the purely ATPG-based approach encompassing the application of 32 precomputed patterns.

The tester merges predefined sequences and pseudo-random patterns generated online. Moreover, it can activate the on-chip logic BIST to improve the toggle activity metric

to 94.68% and the neighborhood-oriented static stress metric to 83.19%.

By exploiting on-chip memory availability to store 12 patterns, the toggle coverage finally reaches 95.8%, and the neighborhood-oriented static stress metric climbs up to 83.74%. The proposed approach demands very limited memory resources. The application for software-based pseudo-random pattern generation is a bare-metal application without any operating system. It corresponds in terms of code size to around 3KB, which are split in the software implementation of a Linear Feedback Shift Register (LFSR), of a Multiple Input Shift Register (MISR), and in their generation/compression seeds. This value has to be compared with about 5MB required to store 32 ATPG patterns for the same device. Memory demands grow to around 160KB memory requirement with each ATPG pattern.

The paper is structured as follows. Section 2 provides some background about Burn-In requirements, testers, and pattern generation. Section 3 illustrates the proposed approach, including the low-cost tester architecture description and the stress effectiveness measurement strategy. Section 4 reports comparative results with an ATPG-based approach demanding a specialized tester. Section 5 draws some conclusions.

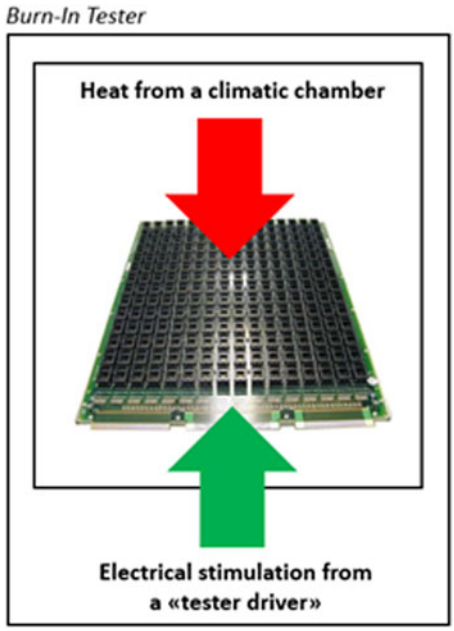


Fig. 1. A Burn-In Board (BIB) and different types of stress that a Burn-In tester can provide.

2 Background

This background part provides the reader with all the needed information to understand the next sections. In particular, it provides descriptive details about the standard Burn-In flow, the tester architectures, the stress evaluation metrics, and the stress generation methods.

2.1 Burn-In Principles

Burn-In [15] uses special equipment, usually called *BI testers*. Such testers extensively use parallelism to reduce test time, resorting to the so-called *Burn-In Boards* (BIBs); a BIB may host up to a hundred packaged chips, and the BI tester's climatic chamber may host many BIBs.

The Fig. 1 shows a single BIB lodged in a Burn-In tester; a production BI tester can host up to tens of such boards.

A BI tester can provide external stress to a large population of devices at a time. The tester drivers stand out of the climatic chamber. They include the circuitries needed to supply electrical stress to the DUTs, possibly at regulated voltage levels (higher or lower than the DUT nominal one, according to technology parameters). The combination of high temperature and controlled voltage is crucial to produce the aging acceleration factor for infant mortality screening, i.e., for exacerbating defective behaviors in chips that would fail shortly after being put into use.

The BI tester can also provide electrical stimulation through the device pins. The tester drivers stimulate the devices "logically" by making them internally exercised while externally stressed. In many cases, this goal is achieved by acting on the DUT scan inputs, continuously fed with suitable ATPG-generated patterns by the BI tester. The following subsection discusses how the ability to internally stress the DUT heavily depends on the activity generated by the applied logic values provided by the BI tester drivers to the devices undergoing the Burn-In stress.

The Burn-In test phase may last for hours, up to 12 if the technology is "young" and/or showing a low maturity level. For a mature product, the duration of the BI reduces to a few hours (typically from 3 to 4 hours). This factor is an important point to consider when evaluating the stress abilities of a generated BI pattern.

2.2 Stress Evaluation Metrics

The availability of effective metrics to measure stress in a DUT is crucial to selecting and possibly improving the stimuli applied to DUT during BI. Many papers [16][17][18] deal with evaluating the stress capabilities of pattern sequences. Industry and academy agree that a stress pattern should be able to activate all circuit nodes, meaning that the pattern set should force '0' and '1' logic values to all gates and nets of the device. A stress pattern unable to toggle all nodes cannot exacerbate all possible manufacturing latent defects, such as weak metallization or oxide imperfections. Therefore, the primary parameter usually evaluated is the so-called *toggle coverage* that measures how many gates moved from '0' to '1' and back at least once during the pattern execution. Another stress measure is the so-called *toggle activity*, which counts the number of toggle events per gate in the circuit. Uniformly distributing the toggle activity is also important because it ensures that all nodes are similarly stressed, but this is considered a secondary goal for an essential stress ability evaluation.

To complete the overall scenario evaluation, the reader should consider that along Burn-In, the same stress patterns run up to millions of times. Therefore, a gate that switches just once per pattern application will switch millions of times throughout the BI process. With the advent of new technologies, more complex activation methods may be required (e.g., static electrical stress approaches, which share some similarities with the IDDQ [19] patterns).

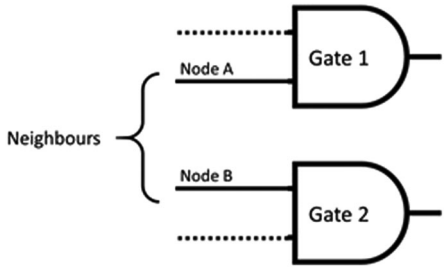


Fig. 2. Neighbours nodes.

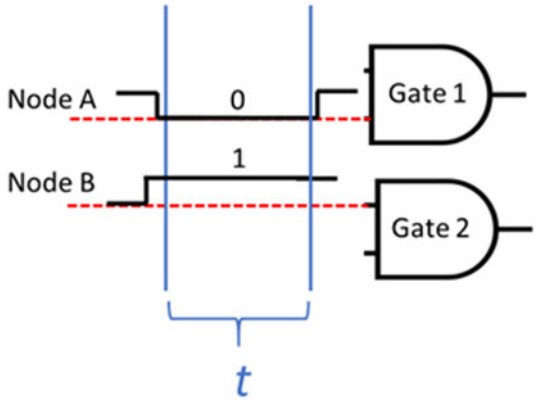


Fig. 3. Neighbour nodes reaching and keeping opposite values for a time *t*.

It is also important to consider the physical layout of the DUT for such a stress evaluation process. For example, by looking at the placement of the gates inside the DUT, it is possible to detect nodes that are neighbors, as shown in Fig. 2. Two lines are defined as *neighbours* if their distance is lower than a given distance. This information permits the creation of a list containing couples of neighbour nodes.

Given a couple of neighbour nodes, intense stress is applied if they keep opposite logic values for a sufficient time *t*. This kind of behavior causes electromagnetic phenomena involving the two nodes [2][20][21][22], exacerbating latent defects.

Therefore, a couple of nodes is fully stressed when both possible configurations with opposite logical values are held. Assuming that the BI tester can access the DUT scan chains, as Fig. 3 depicts, the static stress is applied by keeping "blocked" a specific scan configuration for the specified time t after the initial shift was performed at the tester clock speed. Such a clock is kept stable at the logic value '0' to maintain a static state for a defined amount of time. We named such a measurement as *neighborhood-oriented* static stress metric [14].

2.3 Stress Patterns Generation and Tester Architectures

As it is well known, testers are expensive for a number of reasons, including the requirements in terms of computation power and memory size [23]. This is directly connected to the pattern generation phase and to the mechanisms used to store them before the test phase. Therefore, researchers tried to find out the best trade-off between performances, costs, and test time across pattern generation and test application in order to reduce the tester complexity while increasing its flexibility.

The pattern generation process and in some cases the adopted DfT solutions, such as Logic BIST [24] directly impact the testers' complexity [25]. Both input stimuli and expected output values need to be stored in large memories [25]. Therefore, a clever, flexible, and simple pattern generation drastically relieves the testers' complexity and price. It is crucial in order to further reduce the cost of the final products [25]. Instead of relying on ATPG-generated patterns, *LFSRs* have been widely used for generating pseudo-random patterns. Therefore, either software [26][27] and hardware [28] implementations have been proposed with some approaches not directly related to testing purposes. However, they come with pros and cons related to every implementation in hardware versus. software, i.e., flexibility, speed, and power consumption.

The capabilities and limitations in storing patterns drastically impact the testers' complexity. In the literature several tester architectures have been presented, especially for overcoming memory limitations [29] [30][31][32]. They can be categorized in *low-cost testers*, which can communicate with the DUT, *functional testers* [33][34] that load functional programs, and *BIST solutions*, with random [35], or pseudorandom pattern generation capabilities [36]. However, it is evident that each of the previously mentioned solutions from the literature introduces some hardware overheads, stemming from the introduction of different additional structures ranging from an external FPGA-based tester to modified DfT circuitry within and outside the DUT. This implies additional costs in terms of silicon and adjustments in the manufacturing flow.

Moreover, FPGA-based testers raise the need for a preliminary design phase and its verification. As a consequence, verifying the interconnection from the FPGA to the DUT is another overhead. These preliminary steps indirectly delay the time to market of new devices, possibly impacting the overall costs.

Flexibility and autonomy of testers are also crucial factors in the manufacturing test flow. *Flexibility* allows the reuse of the same testers across different devices of the same family, or even different families. *Autonomy* relieves the test engineer from constantly monitoring testers. Capabilities in terms of maximum pattern application speed are also important. They directly impact the execution time of the test phase and the number of applied patterns.

A more detailed comparison between the testers in the literature and the proposed approach can be seen in Table 1.

3 Proposed Approach

This paper highlights the architecture of a tester environment suitable to provide internal stress (as required by Burn-In test) to large automotive System-on-Chip devices under test (DUTs), while also being compliant with the requirements of System-Level Test (SLT). Indeed, the proposed tester requires a microcontroller unit (MCU) incorporating at least two CPUs and some peripheral modules, like timers, Direct Access Memory (DMA), and I/O modules. The aforementioned architecture is used in this work for stimulating the scan chain(s) and other DfT solutions, but it

TABLE 1
Comparative Evaluation of Different Testers With the Proposed One

Tester	Description	Flexibility	Host PC required	Max Pattern feeding speed	Pattern Memory requirements	Hardware requirements
[29]	Hybrid approach impacting both the external tester and the internal BIST	no	no	NA	ca. 1.5KB	DfT alteration
[30]	FPGA-based with data compression	yes	yes	50MHz	5MB	FPGA design
[31]	FPGA-based with on-board power supplies	yes	yes	NA	NA	FPGA custom board
[32]	FPGA-based applying simulation- extracted patterns	yes	yes	6MHz	576K	FPGA design
Proposed approach	Microcontroller-based	yes	yes	10MHz	ca. 6 MB	CPUs, DMAs, Timers

For a Fair Comparison, Only Structural-Based Testers are Presented.

could also implement a SLT-oriented tester, e.g., to communicate functionally with the DUT.

The proposed tester architecture not only owns the ability of traditional BI testers to apply stimuli stored in tester memory resources, but also exploits the computational ability of the MCU to generate on-line patterns. In more detail, the CPUs included in the MCU execute suitable firmware to:

- Produce a stream of bits to feed the scan chains of the device.
- Read and process the device response.

Nevertheless, the MCU preserves the ability of traditional BI tester to apply a predefined sequence of bits. This ability is important for example to access the test control unit for opening the scan ports (*test mode entry*). This is crucial to access DfT modules and program them to perform stress actions.

Specifically, this work describes how to exploit CPU functionalities to produce a pseudo-random sequence of bits, how the peripheral cores of the microcontroller apply the input stimuli to the DUT and how they read its responses. The CPU firmware performs pattern generation and manages the stimulation results' compression, similar to [37]. The proposed methodology provides a drastic reduction in memory storage demand. A traditional approach only applies to the DUT the pre-computed ATPG patterns, which require up to several MBs of storage memory space in the tester architecture. This is a limitation that often impacts the reachable coverage during BI. For example, in case only 32 scan patterns can be stored, the coverage could be limited compared to the maximum reachable value.

In the proposed architecture, the MCU firmware occupies just a few KBs and can produce an unlimited and carefully customized stream of pseudo-random bits sequenced to the DUT. Besides pure saving memory costs, the improved flexibility is a positive consequence of using an MCU. Such flexibility enables the implementation of composite sequences, for example, generating pseudo-random patterns by programming different generation polynomials and using different generation seeds. Additionally, the MCU firmware can merge predetermined pattern sequences with pseudo-random streams.

In light of all these considerations along the paper, the paper provides the following main contributions:

- It describes the architecture of a tester suitable to supply Burn-In electrical stress to DUTs through its scan chain(s).
- It illustrates how to structure the firmware of the tester by using several CPUs to:
  - Enter a specific test mode
  - Generate a pseudo-random bitstream
  - Sequence the generated patterns to the DUT at medium-high frequency
  - Compress results coming from the DUT.
- It explains how to reach a sufficient stress coverage level by a regular flow that just exercises the chip in a single scan chain test mode and how to complement the coverage by:
  - Adding specific sequences to trigger the available Design for Testability (DfT) modules located on-chip for producing stressful activities, possibly addressing circuit parts that are poorly covered by the pure scan approach.
  - Adding randomization in the use of DfT module, too, e.g., pseudo-random seeds are generated for Logic BIST modules located on-chip.
- It illustrates how to accurately create a set of selective ATPG patterns for improving the stress metrics coverage over a residual list originated by analyzing the pseudo-random stress results.

Fig. 4 illustrates the several steps of the generation method enabled by the tester architecture. In order to effectively access the DUT, the tester can use different test mode entries and DfT activation sequences. These sequences are merged with pseudo-random bits produced on the fly by the tester and the resulting stream sequenced to the DUT. All these operations driven by the tester can be graded in terms of stress ability. The test engineer in charge of preparing the BI recipe can decide to add pseudo-random stress patterns until a satisfactory stress coverage is reached or it stabilizes to a limit. The stress ability is evaluated incrementally along the generation process, leading to a residual list of nodes used as input by a very selective ATPG generation of extra patterns up to the maximum allowed patterns in the leftover space in memory.

3.1 Multicore Tester Architecture

The proposed tester architecture is based on an MCU architecture that includes two CPUs, embedded memory resources,

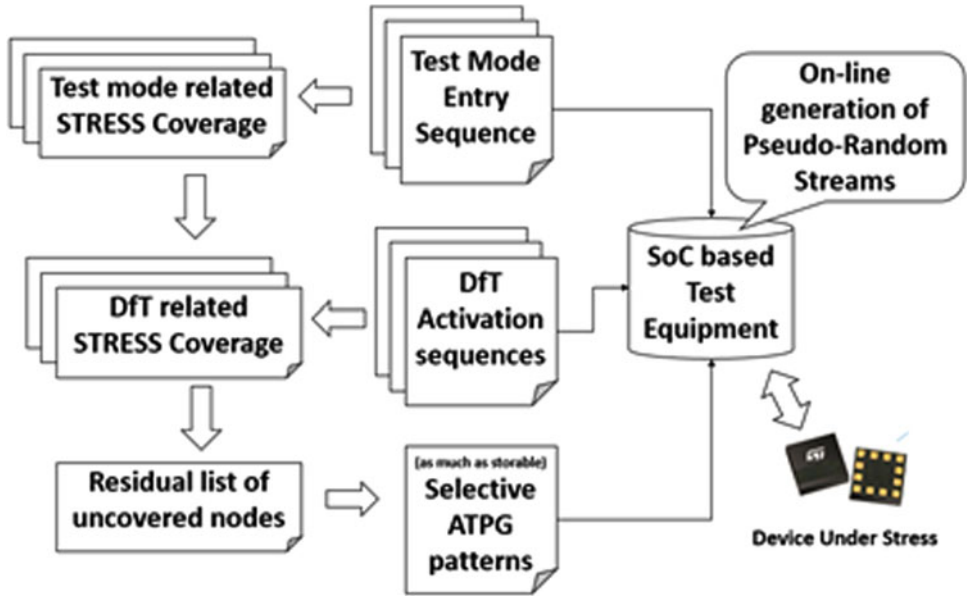


Fig. 4. Overall view of the proposed flow.

and standard peripheral cores, such as DMA controllers, timers, and SPI modules. By orchestrating these modules, we propose a test equipment architecture capable of effectively and adequately stimulating the DUT pins in a functional or DfT-related (e.g., based on scan chains or based on Logic BISTs) mode. The MCU is the most significant component of the tester architecture, and it works under the control of a host PC, which sends high-level commands and reads back the final results of the stress operations. A host PC may drive more than a single microcontroller board, thus allowing to achieve a sufficiently high level of test parallelism [10]. The proposed tester architecture can sequence bits to the pins of the DUT. The tester's CPUs orchestrate this operation with the support of a timer, a DMA controller, and RAM buffers.

Fig. 5 shows the basic architecture of the proposed tester. A pattern set, which could have been created off-line or on-line, is stored into the tester RAM input buffer to be successively sequenced. The pattern values are fetched from the buffer by the DMA controller and then directly moved to the I/O pins. The DMA transfer rate is controlled by a timer that triggers a CPU interrupt every time the input buffer's data has been sent to the DUT to inform the CPU that a new transfer can start. Similarly, pattern application results are collected in the output buffer. The frequency of the interrupt events triggered by the timer depends on the DMA transfer rate. Therefore, the timer counter value depends on the MCU characteristics. If the DMA port is directly connected to a timer output, this handshake protocol among MCU-independent components is further simplified.

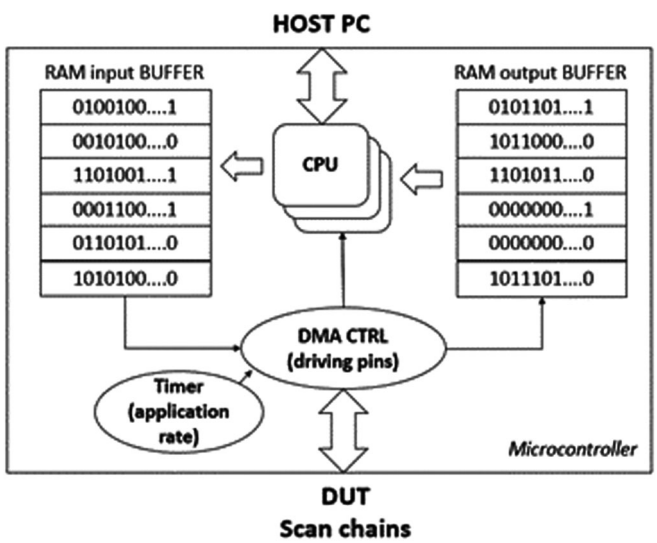


Fig. 5. General overview of the basic test equipment architecture using a microcontroller.

The proposed method is flexible and permits to simultaneously drive many pins; it is also possible to read many output pins of the DUT at the same time. These abilities are suitable to manage multiple scan chains or to support functional stimulation/observation, as also required by SLT.

The above ability is also crucial because of security constraints. A prologue sequence of predefined bits (including cryptography keys) is needed to enter a so-called "test mode" status. This special configuration is accessible by programming a set of suitable registers of a Test Control Unit inside the DUT through a TAP controller. The test mode functionalities are indispensable to set up the scan chain(s) and test clocks. Similarly, the DfT module can be activated by specific sequences of bits that trigger self-test functionalities. Both test mode and DfT-related sequences can be stored in the RAM buffer and sequenced at any time when needed.

Furthermore, the presence of CPUs opens to on-chip intelligence. As detailed in the next subsection, the CPUs can produce a stream of bits stored in the available input buffer by firmware. This stream can be fed to the device under stress once it enters a test mode or feeds the DfT modules with seeds for self-test application. The CPUs can also elaborate on the resulting output bits accumulated in the output buffer. Such ability to intervene in the stream creation also permits to produce very well-tailored sequences to merge fixed bits with pseudo-random ones produced on the fly. When using different DfT techniques rather than the

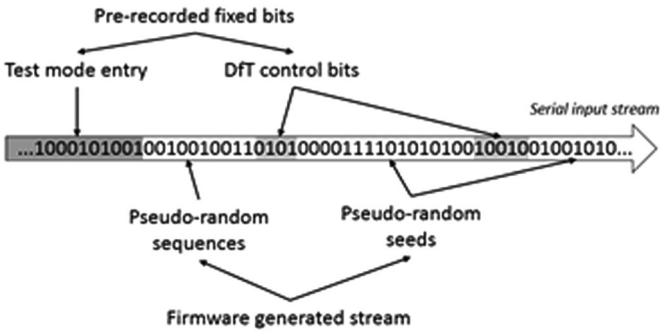


Fig. 6. Example of produced stream of bits by combining fixed, pre-recorded bits with variable sequences created by the tester intelligence.

scan mode, such as internal Logic BISTs or compressors, this ability becomes very beneficial.

An example is shown in Fig. 6, which could be the case of a Logic BIST. Similar to every other stress sequence, the first bits are needed to activate the test mode. Such test mode entry comprises fixed bits to be stiffly applied before starting the real stress application. If a DfT module is used after the test mode entry, this will require a combination of information to be received from the tester. For the LBIST case, seed and control bits are sent alternatively. Control bits are pre-recorded, while seed bits are generated on the fly.

In this context, some extra considerations concerning the test application frequency are needed, even though the shift frequency is less relevant than the high internal clock frequencies activated during the test procedures. A major challenge when devising the tester architecture is how to reach the maximum test frequency while sequencing the bits composing the pattern.

This objective is quite easy to achieve when the pattern is predetermined and stored in the internal memory; if the pattern is generated on the fly by the software running on the tester CPU, then additional efforts are required to match the frequency expectations. In other words, if the firmware running on the CPU is too slow in producing the pseudo-random patterns, then the DMA is asked to wait until new bits are available, creating a discontinuity in the pattern application. Therefore, the quality of the tester firmware is crucial, as well as the decisions about using embedded peripherals to supply GPIO pins with DUT inputs.

For example, let's consider the most common Burn-In Test mode that permits the usage of a single, full scan chain reaching all FFs in the circuit. The scan chain can be used in such a basic test mode after the entry sequence is completed. At this point, the chain can be controlled by shifting and capturing values through regular scan signals scan In (SI), Scan Out (SO), and Scan Enable (SE), other than the Scan Clock that is mapped on pins. For this case, both CPU computational power of the tester and on-chip I/O peripherals can be exploited to speed up the stimulation and reduce the CPU workload. A lower CPU workload permits producing more bits in time, while a high workload means stalling the stimulation and waiting for new stream bits to be produced.

The SPI peripheral module is an example of a proper interface to access the single full chain. As shown in Fig. 7, the resulting setup when adding I/O peripherals like the SPI is similar to the basic one described previously, with DMA and timer involved again; additionally, the SPI

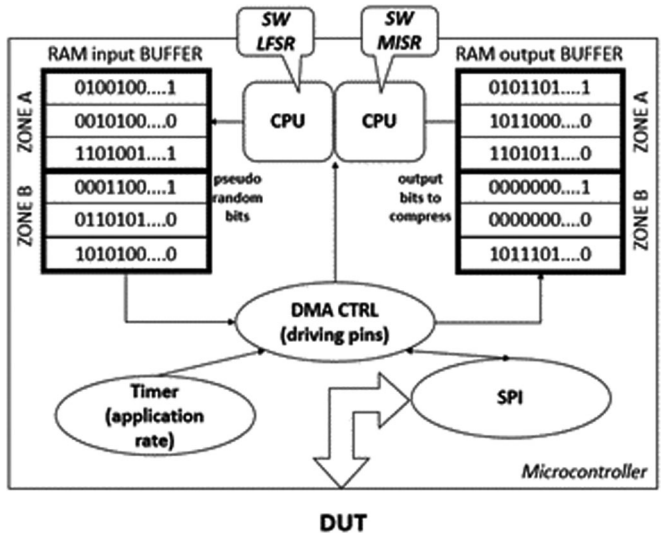


Fig. 7. Proposed test equipment organization to autonomously generate and apply pseudo-random patterns.

module is used to send bits to the DUT instead of the DMA after the test mode entry sequence is completed. Using the SPI is a winning solution because it autonomously provides the clock signal, whose value is not stored in the input buffer.

In summary, by exploiting the I/O features, it is possible:

- To further reduce the amount of memory space required to store the generated pattern.
- To minimize the amount of data transferred through DMA to I/O peripherals.
- To lighten the CPU workload for creating patterns.
- To overall increase the patterns fed to the DUT.

The fastest ports are always preferable to ensure faster stimulation, and the SPI ensures quite high frequencies (up to more than 10MHz). The stimulation bits that the CPUs on the fly produce through the SPI port are sent to the scan input. More in detail, the CPU produces bits and stores them in the RAM input buffer, while the DMA fetches data from the input buffer and sends them to the SPI peripheral for stimulating the DUT. In order to avoid conflicts between the CPU generating bits and the DMA fetching them, the RAM input buffer is divided into two zones: while the CPU is filling Zone A with the produced bits, the DMA fetches patterns from Zone B and vice-versa.

Similarly, the RAM output buffer is divided into two zones; the DMA uses one to deposit the read values from the DUT, and the CPU can access the other to compute any signature.

This well-balanced system permits continuous stimulation without any stall or intermittent stimulation if the CPU is fast enough to fill a zone right before the bits in the opposite are all sequenced. We propose to use multiple CPUs, if available, to light up the duties of every single one and ensure a sufficient generation rate. In our scenario, we used a CPU for creating a sequence of bits by a software implementation of a Linear Feedback Shift Register (LFSR). Another CPU is computing a signature out of the collected output bits through a Multiple Input Shift Register (MISR), also implemented in software.

The MISR module can provide some information about DUT failures along with the BI phase. Also, it is useful to ensure that the chip is still connected correctly and the communication is not lost during the hours of BI stress application[7]. This ability is crucial to ensure the proper time of electrical stress during BI.

Moreover, any other software generation method can be used to replace the popular LFSR and MISR modules used here.

3.2 Pattern Generation Flow

A major objective for the illustrated tester architecture is to reach the same level of stress coverage supplied by traditional BI testers without demanding large memory resources. A test engineer that would move from traditional, ATPG-oriented BI testers to the proposed one has to select a recipe that is not coming from an automatic pattern generator.

In order to select the right kind and amount of patterns needed to reach the same stress coverage by a traditional BI tester, a simulative approach [14] can be used to compute a stress coverage value. Obviously, this simulation step requires the DUT netlist and a proper simulation environment.

The stress method must ensure the usage of the test modes enabling broader access to the device logic to achieve a sufficient stress coverage. It is strongly suggested to use a single full scan architecture as the starting point because it provides an easy solution with a low pin count, and it powerfully stresses the majority of the circuit nodes.

A key point of the proposed tester structure is the capability of the generation engine to create a larger pattern set than any other that can be memorized in the MCU embedded memories. The number of patterns created is the key to ensuring high coverage. For example, if the toggle activity is considered as the stress metric, the ATPG may generate a relatively small pattern set. Using alternative generation techniques such as pseudo-random ones it is usually possible to reach the same stress level but with a much larger pattern set[38].

Fig. 8 shows a simple pseudo-code illustrating how to proceed in the selection of the cardinality of the generated pattern set. In addition, changing from one Test Mode to another often means switching the DfT modules.

As soon as the illustrated process has ended, a so-called *residual list* of nodes is also available, which contains the list of not toggled nodes. A set of complementary patterns can be created through the traditional ATPG approach by targeting nodes in the residual list.

A balance needs to be found between the number of pseudo-random patterns generated on the fly at test time and the number of patterns created off-line resorting to an ATPG, considering the limitations imposed by the selected microcontroller architecture. In particular, the memory space available on the tester is a strong constraint to be considered.

4 Experimental Results

This section reports the results obtained by resorting to hardware and software implementations of the proposed architecture. The considered target Device under Stress is a 40nm Automotive Microcontroller [39], equipped with configurable scan chains and all features to perform a BI stress by operating on the scan inputs. Overall this device includes about 20 million circuit nodes in the logic parts and 700K

```
ATPG = referenceCoverage
Coverage = 0
do{
        do{
                add patterns
                Coverage += patternsCoverage
        } while Coverage < ATPG or in steady state
        change TestMode
} while TestModes != ENDED
FinalCoverage = Coverage
```

Fig. 8. Simulation-based algorithm to control pattern generation on-chip.

Flip Flops in the scan chains. The SoC includes many functionalities like processor cores connected to peripheral cores and embedded memories.

The most important results achieved are:

- A proof of concept corresponding to an MCU-based tester with good performance (up to 10MHz sequencing rate with non-optimized interconnections between MCU-based tester and DUT).
- The observation that a pseudo-random stream of bits generated on the fly by the tester provides the same (or better) stress level than ATPG-generated patterns applied by traditional BI testers.

For this specific case of study:

- The pseudo-random generation is based on a software implementation of a 32-bit wide LFSR generator and a MISR with similar characteristics.
- The comparison in terms of stress coverage is done with the 32 ATPG pattern set sequenced to the DUT during current volume production by a traditional BI tester.

Many tester architectures and pattern compaction techniques have been presented in the past years, as explained in Section 2.3 and described in Table 1. A fair comparison between the proposed approach and those in the literature is difficult to provide. In fact, most of the state-of-the-art testers need ad-hoc circuits and/or a companion FPGA to work correctly, while the proposed method aims to provide a tester architecture without additional hardware circuitry and FPGA. The simplicity of our approach makes it innovative concerning previous implementations, which demand a costly design phase.

4.1 Tester Implementation, Evaluation and Performance

A development board by STMicroelectronics was used to implement the tester concept. This board is based on a multi-core system-on-chip with several peripheral cores, running at 180MHz. In our case of study, the CPUs drive the I/O pins of the SoC connected to the DUT, as in Fig. 10. The firmware is a bare-metal application, commanding the CPU to generate the test sequence. In more detail, in the inner part of the firmware, an ASM-coded function is in charge of generating the sequence of pseudo-random patterns mimicking in software a 32-bits LFSR. The ASM function returns the next bit to be shifted in the chain; this bit is temporarily stored in buffers, whose behavior is explained in Section 3.1. The buffer content is then sequenced to the DUT at a maximum frequency of 10MHz.

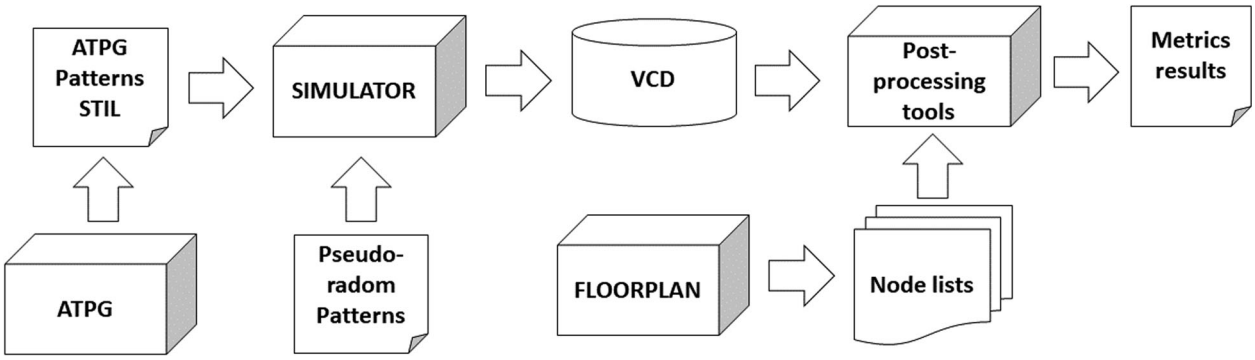


Fig. 9. Toolchain for stress metrics evaluation.

By leveraging on the multi-core architecture of this companion chip, output compression is assigned to an alternative CPU different than the one used for creating the input stream. Such distributed approach permits to maintain the 10 MHz external application frequency and may enable further gain if a more aggressive technology is used, e.g., a PCB card with shielded wires instead of normal wires connected through a breadboard.

Code size is quite small, reaching 3 Kb with about 1,300 code lines, 50 of them written in assembly language, while the others are in C. Besides the code size, storing a single ATPG pattern for the DUT requires about 160KB of memory in the tester architecture.

4.2 Stress Activity Evaluation Results

In order to provide effective stress, we have followed the strategy described in Section 3.2. Our efforts concentrated on two stress metrics:

- The Toggle Activity metric, measuring how many nodes out of those in the whole circuit did toggle at least once.
- The neighborhood-oriented static metric, considering a set of couples of neighbour nodes (the distance for considering two nodes close to each other is 6 μm) and the states these couples hold during the stimulation (as described in Section 2.2).

Fig. 9 displays the flow implemented for computing the listed metrics. A toolchain including ATPG, Simulation, Floorplan tools, and post-processing applications is used to grade ATPG and pseudo-random patterns. More in detail, we used the NCSIM tool from the Incisive Suite (Cadence), the Tessent ATPG (Mentor), and the Innovus tool (Cadence). As it is very well known, exhaustive simulation of scan-based stress patterns is very time-consuming (up to months). Therefore, the deductive approach described in [14], is used to parallel load in all scan registers the stress patterns for speeding up the simulation by several orders of magnitude. A parallel engine developed in the frame of the project by the Politecnico di Torino[40] analyzes the simulation VCD file. ATPG tools usually return some stress coverage as a result. In our case, we used the fault simulation capabilities of the selected ATPG fault simulator to perform a regression of our system based on the pure toggle coverage. Of course, ATPG and fault simulation techniques could not be used to perform extended statistics and neighborhood-oriented stress metrics.

We considered the activity produced by a 32 ATPG-based pattern set for all the performed measurements. This pattern set is taken as a reference value because it is the exact pattern set applied during the volume production Burn-In phase. Such a low number of patterns is due to the very long scan chain of almost 700K FF, requiring quite a lot of memory space per pattern. Results provided by the ATPG are validated and compared with many sequences composed of a growing number of patterns.

On the whole SoC, ATPG patterns reach up to 90.73% and 81.05% of toggle activity and neighborhood-oriented static stress, respectively. With a very large set of pseudo-random patterns, the figures produced by our method are better than the ATPG results. The results obtained by the pseudo-random patterns are very similar to those obtained with ATPG ones. The pseudo-random approach is composed of more vectors than the ATPG approach, but it does not require extra memory since it is produced by the tester CPU running the 3K large generation code. Conversely, memory limitations do not increase the number of ATPG-created patterns.

For the sake of completeness, we left an ATPG process generating an unlimited number of patterns, and imposing no time constraints. The returned value could be considered as the absolute reference of how high could be the toggle coverage. This experiment returned a maximum toggle coverage of 94.13% over the 20 millions node of the circuit with 327 patterns.

In Table 2 we report the results obtained when evaluating the stress coverage of 32/128/1,024 pseudo-randomly generated patterns. It is worth mentioning that the larger number of vectors produced by the pseudo-random approach does not represent an issue here since the whole BI test lasts for hours, during which the same patterns are repeatedly applied many times to the DUT.

Regarding the distribution of the nodes forced to toggle by the ATPG and the proposed approach with the software LFSR, Fig. 11 clarifies the relationships among the set of nodes toggled by the different methods over the entire SoC. It results that the pseudo-random one stimulates some nodes in the circuit (3.14% of the total) that the ATPG is not able to toggle. On the other hand, the ATPG activates some nodes (1.69% of the total) which are not toggled by the pseudo-random pattern.

Even though the main goal in the present work is to toggle a node at least once, we have also evaluated how the activity is distributed. The average number of toggles made by each node with the ATPG patterns results is 33.54, while for the

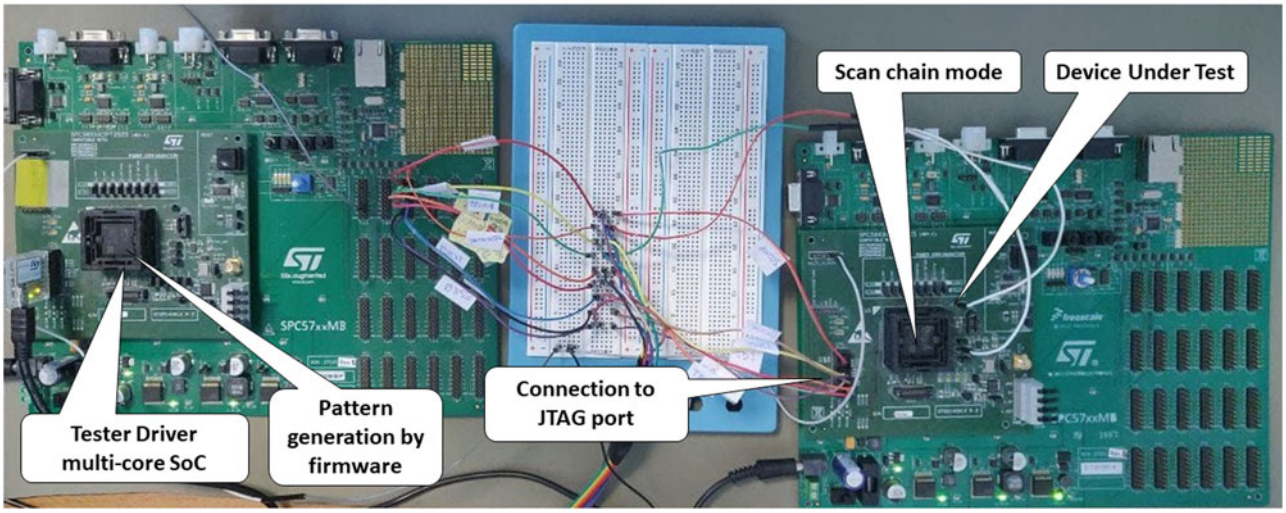


Fig. 10. Experimental setup.

pseudo-random patterns is 32.38. By comparing the total amount of toggles created by the two approaches, they perform similarly: the ATPG-generated patterns cause about 666 million toggles, while the pseudo-random patterns case about 663 million toggles. In a Burn-In process running for 3 hours, the pattern is continuously and cyclically applied. This means that the pattern set is applied about 50,000 times. Hence, every node is guaranteed to toggle at least a million and a half times. Fig. 12 illustrates these results.

In order to possibly identify the sources of missing coverage, we also plotted the coverage over the device layout. This kind of visualization of the stress is shown in Fig. 13, where the green areas are well stressed, while red spots are related to poorly covered regions of the device. Such a visualization approach permitted identifying some DfT-related areas that were not covered or poorly covered. This observation triggered the selection of the Test Modes to be exercised other than the single full scan scenario.

In our specific context, we decided to activate the Embedded Deterministic Test Xpress [41][42][43][44] system, which hosts an LFSR-based decompressor and a combinational compactor with X-masking capabilities. Each compressed pattern is composed of three parts: a decompressor reset sequence, the actual pattern (which includes some configuration bits for the X-masking circuit), and a "post-shift" sequence in which the capture circuitry is activated to perform launch-on-capture. The reset sequence is the same for every pattern, while the post-shift sequence differs from one pattern to another.

We considered the results reached by an ATPG-generated pattern set, targeting the transition delay fault model as coverage reference for such a DfT method. Around 200k previously uncovered nodes were made toggling by this DfT-based pattern. Experiments have been then performed on patterns obtained by modifying the seeds of the ATPG-generated compressed patterns and replacing them with pseudo-random values. The results closely follow the ones obtained by the ATPG patterns. With the generation of 18 DfT-based patterns, the updated toggle and neighborhood-oriented static stress metrics coverage reach 94.68% and 83.19%, respectively.

As a final step, given the memory resources of the test driver, we generated some selective ATPG patterns over the residual list of all nodes not toggled by the generated pattern set, i.e., LBIST-based, 32 ATPG, and 1,024 pseudo-random. The graph reported in Fig. 14 shows the progression of the incremental coverage over the residual list. Interestingly, an essential jump in the coverage is obtained with just one selective ATPG pattern, while the curve tends to saturate after ten patterns.

The residual coverage trend (in blue) can be compared with the red line showing the coverage evolution of the unconstrained ATPG generation experiment mentioned before. In this case, the ATPG reaches 90% with the 32 patterns that the traditional BI tester can store. From 90% to the final 94%, the trend is very similar to the selective ATPG generation. The curve grows fast up to nearly 94% with about ten additional patterns. Afterward, the curve slowly grows to reach the final 94.13%. There is about a 1%

TABLE 2
Toggle Coverage Results

Pattern type	Number of test vectors	Toggle activity 19,651,246	Static stress 20,755,132
		nodes	couples
ATPG	32	90.73%	76.97%
Pseudo	32	86.36%	72.42%
random	128	89.31%	77.04%
	1,024	92.18%	81.05%

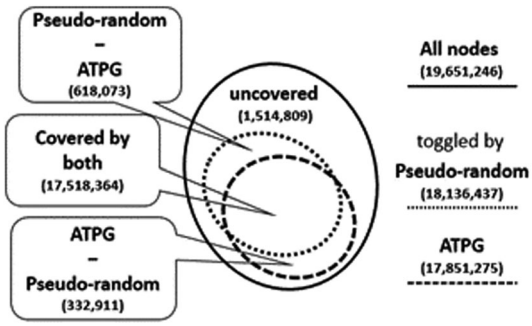


Fig. 11. Toggle coverage comparison between 1,024 pseudo-random and 32 ATPG-generated patterns.

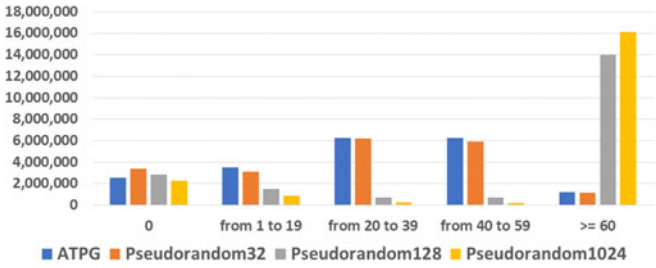


Fig. 12. Distribution of the number of toggles per node.

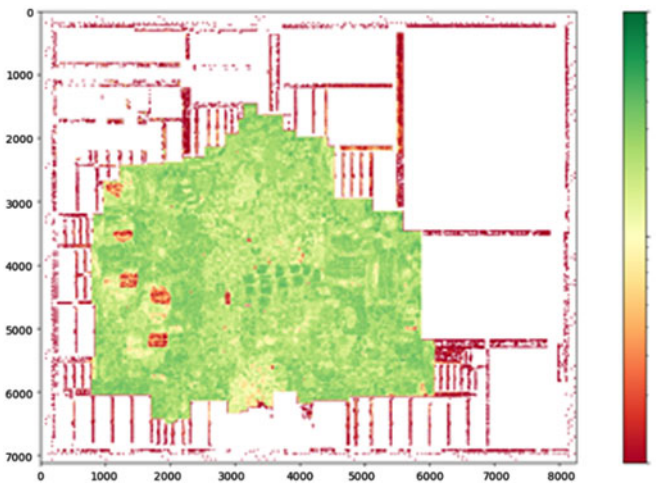


Fig. 13. Visualization over the layout of the toggle coverage produced by pseudo-random patterns used to excite the single full scan chain.

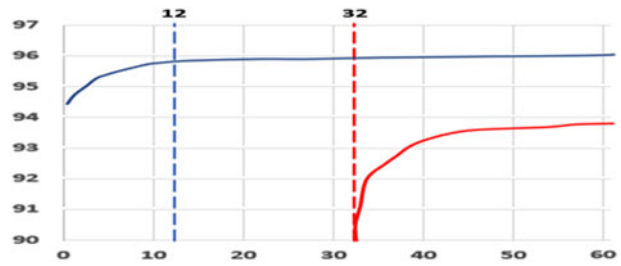


Fig. 14. ATPG-based generation of additional patterns and the achieved toggle coverage.

coverage gain by the proposed method, which is obtained by the pseudorandom seed-based LBIST stress.

Every pattern requires about 160KB to be stored without any compression. In the adopted architecture, the microcontroller can store only one selective ATPG pattern in its embedded SRAM and up to 12 patterns in the embedded 6MB FLASH core.

With these memory constraints, the final stress coverage figures on the entire SoC, reached by adding 12 selective ATPG patterns, are 95.89% for the toggle coverage and 83.74% for the neighborhood-oriented static stress.

Table 3 summarizes the evolution in terms of measured stress metrics. The incremental superposition of different approaches allows reaching a very high toggle activity level of 95.89%.

Fig. 15 displays the final coverage reached summing up all stress contributions,i.e., on-chip activation of LBIST, selective ATPG patterns, 1,024 pseudo-random and 32 ATPG. It depicts the strength of superimposing different

TABLE 3
Overall Cumulative Toggle Coverage Results

Pattern type	Number of test vectors	Toggle activity 19,651,246 nodes	Static stress 20,755,132 couples
Pseudo-random	1,024	92.18%	81.05%
+ LBIST-based	18	94.68%	83.19%
+ Selective ATPG	12	95.89%	83.74%

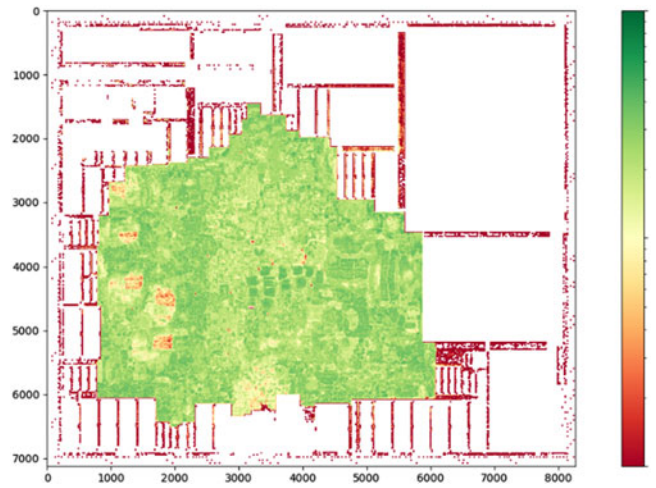


Fig. 15. Visualization over the layout of the toggle coverage produced by all stress activities driven by the tester.

stress approaches to improve the overall stress in the device. For example, comparing Figs. 15 and 13 the not toggled nodes in red have been stimulated completely, or partially for some parts of the device. On the other hand, some poorly stimulated nodes, in yellow, in Fig. 13 have been completely stimulated in Fig. 15.

The missing coverage, which is about 5% for the entire SoC, is quite distributed over the SoC elements. With a deeper analysis we classified these nodes as very hard or impossible to toggle nodes utilizing structural methods, such as scan and LBIST DfT schemes.

Very hard nodes may belong to modules executing very composite computations, which are difficult to target by their nature; in our case, some decimals of missed coverage are related to the SoC embedded timers.

Impossible to toggle nodes are often related to connections to memory cores, which appear in red in the visual display shown in Fig. 15. Such nodes can be covered by alternative approaches like Functional program execution, as it happens for System-Level-Test oriented strategies [45] [8]. The logic parts connecting memories and analog modules and some timer modules that are blocked by the test mode are not toggled by the approach. These components will be exercised along with the dynamic Burn-In phase with functional procedures.

5 Conclusion

This study demonstrates that an MCU-based tester can be used to reach effective stress coverage with limited resources and memory space. The MCU firmware can merge the

generation of pseudo-random and control sequences of bits and produce an unlimited stream of bits. The overall toggle coverage metric reached almost 96%.

The paper describes how to program the MCU firmware such that the generation process is fast enough compared to the volume production tester. In our case, 10MHz is reached by using unshielded wires connected through a breadboard.

The high-stress coverage and the good application frequency reached by the MCU make it feasible to apply electric stress during other test steps where the tester is based on MCU, i.e., the system level test phase.

REFERENCES

[1] E. Armengaud, A. Steininger, and M. Horauer, "Towards a systematic test for embedded automotive communication systems," *IEEE Trans. Ind. Inform.*, vol. 4, no. 3, pp. 146–155, Aug. 2008.

[2] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie, "Temperature-aware NBTI modeling and the impact of standby leakage reduction techniques on circuit performance degradation," *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 5, pp. 756–769, Sep./Oct. 2011.

[3] "International standard - IEC 61508 - functional safety of electrical/electronic/programmable electronic safety-related systems," *Int. ElectroTech. Commission*, 2010.

[4] "Road vehicles – functional safety," ISO 26262, 2018.

[5] A. Benso, A. Bosio, S. D. Carlo, G. D. Natale, and P. Prinetto, "ATPG for dynamic burn-in test in full-scan circuits," in *Proc. 15th Asian Test Symp.*, 2006, pp. 75–82.

[6] C. He, "Advanced burn-in - an optimized product stress and test flow for automotive microcontrollers," in *Proc. IEEE Int. Test Conf.*, 2019, pp. 1–6.

[7] D. Appello et al., "An optimized test during burn-in for automotive SoC," *IEEE Des. Test*, vol. 35, no. 3, pp. 46–53, Jun. 2018.

[8] I. Polian et al., "Exploring the mysteries of system-level test," in *Proc. IEEE 29th Asian Test Symp.*, 2020, pp. 1–6.

[9] D. K. R. Tipparthi and K. K. Kumar, "Concurrent system level test (CSLT) methodology for complex system-on-chip," in *Proc. IEEE 16th Electron. Packag. Technol. Conf.*, 2014, pp. 196–199.

[10] F. Almeida et al., "Effective screening of automotive SoCs by combining burn-in and system level test," in *Proc. IEEE Int. Symp. Des. Diagnostics Electron. Circuits Syst.*, 2019, pp. 1–6.

[11] M. Radu, "Testing digital circuits using a mixed-signal automatic test equipment," in *Proc. IEEE Int. Conf. Automat. Qual. Testing Robot.*, 2014, pp. 1–4.

[12] R. Nielsen and D. A. Tagliente, "Modular automatic test equipment design for on-platform diagnostics," in *Proc. IEEE Autotestcon*, 2015, pp. 181–185.

[13] N. Touba, "Survey of test vector compression techniques," *IEEE Des. Test Comput.*, vol. 23, no. 4, pp. 294–303, Apr. 2006.

[14] W. Ruggeri et al., "Innovative methods for burn-in related stress metrics computation," in *Proc. 16th Int. Conf. Des. Technol. Integr. Syst. Nanoscale Era*, 2021, pp. 1–6.

[15] C. He et al., "Wafer level stress: Enabling zero defect quality for automotive microcontrollers without package burn-in," in *Proc. IEEE Int. Test Conf.*, 2020, pp. 1–10.

[16] M. Zakaria, Z. Kassim, M.-L. Ooi, and S. Demidenko, "Reducing burn-in time through high-voltage stress test and weibull statistical analysis," *IEEE Des. Test Comput.*, vol. 23, no. 2, pp. 88–98, Mar./Apr. 2006.

[17] D. Appello et al., "A comprehensive methodology for stress procedures evaluation and comparison for burn-in of automotive SoC," in *Proc Des. Automat. Test Europe Conf. Exhib.*, 2017, pp. 646–649.

[18] X. Guo, W. Burleson, and M. Stan, "Modeling and experimental demonstration of accelerated self-healing techniques," in *Proc. 51st ACM/EDAC/IEEE Des. Automat. Conf.*, 2014, pp. 1–6.

[19] R. Kawahara, O. Nakayama, and T. Kurasawa, "The effectiveness of IDDQ and high voltage stress for burn-in elimination [CMOS production]," in *Proc. Dig. Papers 1996 IEEE Int. Workshop IDDQ Testing*, 1996, pp. 9–13.

[20] Z. Tokei et al., "Impact of LER on BEOL dielectric reliability: A quantitative model and experimental validation," in *Proc. IEEE Int. Interconnect Technol. Conf.*, 2009, pp. 228–230.

[21] K. Croes, D. Kocaay, I. Ciofi, J. Bömmels, and Z. Tökei, "Impact of process variability on BEOL TDDB lifetime model assessment," in *Proc. IEEE Int. Rel. Phys. Symp.*, 2015, pp. BD.5.1–BD.5.5.

[22] D. Vogel, S. Rzepka, B. Michel, and A. Gollhardt, "Local stress measurement on metal lines and dielectrics of beol pattern by stress relief technique," in *Proc. Semicond. Conf. Dresden*, 2011, pp. 1–3.

[23] A. Crouch, "Future trends in test: The adoption and use of low cost structural testers," in *Proc. Int. Conf. Test*, 2004, pp. 698–703.

[24] R. Bhakthavatchalu, S. Krishnan, V. Vineeth, and M. N. Devi, "Deterministic seed selection and pattern reduction in logic bist," in *Proc. 18th Int. Symp. VLSI Des. Test*, 2014, pp. 1/2.

[25] M. Rehani, D. Abercrombie, R. Madge, J. Teisher, and J. Saw, "Ate data collection - a comprehensive requirements proposal to maximize ROI of test," in *Proc. Int. Conf. Test*, 2004, pp. 181–189.

[26] S. Chowdhury and S. Maitra, "Efficient software implementation of linear feedback shift registers," in *Proc. 2nd Int. Conf. Cryptol. India*, 2001, pp. 297–307, doi: 10.1007/3–540-45311-3_28.

[27] D. Okunbor, "Software implementation of LSFR-based stream ciphers for GSM cryptosystems," in *Proc. 7th Int. Conf. Rel. Infocom Technol. Optim.*, 2018, pp. 59–65.

[28] A. K. Panda, P. Rajput, and B. Shukla, "FPGA implementation of 8, 16 and 32 bit LFSR with maximum length feedback polynomial using VHDL," in *Proc. Int. Conf. Commun. Syst. Netw. Technol.*, 2012, pp. 769–773.

[29] A. Jas, C. Krishna, and N. Touba, "Hybrid BIST based on weighted pseudo-random testing: A new test resource partitioning scheme," in *Proc. 19th IEEE VLSI Test Symp.*, 2001, pp. 2–8.

[30] L. Ciganda, F. Abate, P. Bernardi, M. Bruno, and M. Sonza Reorda, "An enhanced FPGA-based low-cost tester platform exploiting effective test data compression for SoCs," in *Proc. 12th Int. Symp. Des. Diagnostics Electron. Circuits Syst.*, 2009, pp. 258–263.

[31] A. Patel, V. Gosain, R. S. Mohal, and J. Kumar, "FPGA based low-cost portable tester with on-board supplies," in *Proc. 4th Int. Conf. Electron. Commun. Aerosp. Technol.*, 2020, pp. 301–306.

[32] L. Mostardini et al., "FPGA-based low-cost system for automatic tests on digital circuits," in *Proc. 14th IEEE Int. Conf. Electron. Circuits Syst.*, 2007, pp. 911–914.

[33] D. Jadaan, K. Gonsholt, and A. Skavhaug, "Low-cost platform-independent FPGA based real-time systems tester," in *Proc. Euromicro Conf. Digit. System Des.*, 2016, pp. 686–689.

[34] D. Harris and D. Diaz, "Testosterics: A low-cost functional chip tester," in *Proc. IEEE Int. Conf. Microelectronic Syst. Educ.*, 2003, pp. 74–75.

[35] C.-C. Chi, C.-Y. Lo, T.-W. Ko, and C.-W. Wu, "Test integration for SoC supporting very low-cost testers," in *Proc. Asian Test Symp.*, 2009, pp. 287–292.

[36] J.-C. Rau, Y.-F. Ho, and P.-H. Wu, "A novel reseeding mechanism for pseudo-random testing of VLSI circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, pp. 2979–2982.

[37] M. Bakiri, C. Guyeux, J.-F. Couchot, L. Marangio, and S. Galatolo, "A hardware and secure pseudorandom generator for constrained devices," *IEEE Trans. Ind. Inform.*, vol. 14, no. 8, pp. 3754–3765, Aug. 2018.

[38] P. Girard, C. Landrault, S. Pravossoudovitch, and A. Virazel, "Comparison between random and pseudo-random generation for bist of delay, stuck-at and bridging faults," in *Proc. 6th IEEE Int. On-Line Testing Workshop*, 2000, pp. 121–126.

[39] STMicroelectronics, "SPC58NN84C3, 32-bit power architecture MCU for high performance applications," [Online]. Available: https://www.st.com/en/automotive-microcontrollers/spc58nn84c3.html

[40] D. Appello et al., "Accelerated analysis of simulation dumps through parallelization on multicore architectures," in *Proc. 24th Int. Symp. Des. Diagnostics Electron. Circuits Syst.*, 2021, pp. 69–74.

[41] J. Rajski, N. Mukherjee, M. Kassab, T. Rinderknecht, and M. Dessouki, "Scan test application through high-speed serial input-outputs," US Patent US20100313089A1, 2010.

[42] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 5, pp. 776–792, May 2004.

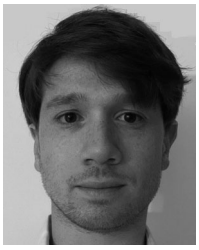
[43] J. Rajski et al., "Embedded deterministic test for low cost manufacturing test," in *Proc. Int. Test Conf.*, 2002, pp. 301–310.
[44] J. Rajski, J. Tyszer, G. Mrugalski, W.-T. Cheng, N. Mukherjee, and M. Kassab, "X-press compactor for 1000x reduction of test data," in *Proc. IEEE Int. Test Conf.*, 2006, pp. 1–10.
[45] H. H. Chen, "Beyond structural test, the rising need for system-level test," in *Proc. Int. Symp. VLSI Des. Automat. Test*, 2018, pp. 1–4.

Francesco Angione (Student Member IEEE) received the MSc degree in computer engineer from Politecnico di Torino, in 2020. He is currently working toward the PhD degree with Politecnico di Torino, in CAD & Reliability group. His main interests include real-time operating systems, computer architectures, and programmable architectures and their reliability.

Davide Appello received the degree in electronic engineering from the "Alma Ticinensis Universitas" of Pavia, in Italy. He is director of product engineering and industrialization for automotive digital products, RF, and Audio at STMicroelectronics, in Italy. He has been with ST for more than 25 years.

Paolo Bernardi (Senior Member, IEEE) received the MS degree in computer science, in 2002, and the PhD degree in computer science, in 2006. He an Associate Professor with the Politecnico di Torino University, where he works in the Electronic CAD and Reliability research group. His current interests include system-on-chip tests and reliability, especially in the direction of high-quality automotive devices. He is the Program Chair of the Automotive Reliability, Test and Safety (ARTS) Workshop held in conjunction with the International Test Conference and General Chair of the IEEE European Test Symposium 2023.

Claudia Bertani received the graduate degree in electronic engineering from Politecnico di Milano. She is a product & test engineer manager with STMicroelectronics. More than 20 years of expertise in the semiconductor industry, high volume products ramp-up and management, semiconductor manufacturing flow, DfT and testability analysis, ATE, new product introduction technical support, and team management. She is currently managing the team in charge of System-Level Test introduction on Automotive ADAS SOC products.



Giovambattista Gallo received the MSc degree in computer engineer from Politecnico di Torino, in 2020. His thesis work done with Politecnico di Torino, CAD & Reliability group treated hardware aspects of Burn-In Testers.

Stefano Littardi received the MSc degree in computer engineer from Politecnico di Torino, in 2019. In 2020, he worked as a research fellow Politecnico di Torino with the CAD and Reliability group, taking care of software aspects of burn-in testing.

Giorgio Pollaccia received the degree in electronics engineering from the Università di Palermo, Palermo, Italy. He is a burn-in test engineer for automotive digital products with STMicroelectronics, where he is involved in testability and testing.

Walter Ruggeri received the bachelor's degree in computer engineering from Politecnico di Torino, in 2018, and the master's degree in computer engineering from Politecnico di Torino, in 2020, and since then has worked as a research assistant for the same institution. His research interests include design-for-testability and new digital devices stress metrics development.

Matteo Sonza Reorda (Fellow, IEEE) received the master's degree in electronics and the PhD degree in computer engineering from Politecnico di Torino, Italy, in 1986 and 1990, respectively. He is currently a full professor with the Department of Control and Computer Engineering of the same institution. He published more than 400 papers in the area of test and fault-tolerant design of reliable circuits and systems, receiving several best paper awards at major international conferences. He is involved in numerous research projects with companies and other research centers worldwide.

Vincenzo Tancorre received the BS degree in electronics engineering from Politecnico di Bari. He is a Yield Enhancement engineer with the Automotive Group, STMicroelectronics. His research interests include test-related process monitoring using diagnostic solutions for memories and unstructured logic based on DfT methodologies.

Roberto Ugioli received the graduate degree in electronic engineering from Politecnico di Torino, in 1990. He gathered thirty years of experience in design for testability and silicon validation of ASIC devices. Since 2018, at STM Automotive Division, Senior Hardware Product Engineer in charge of System-Level Test equipment for volume production tests of ADAS devices.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement